REMARKS

This application has been carefully considered in connection with the Office Action dated October 2, 2008. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 21-24, 26, 28, 30, and 35-37 were rejected under 35 USC § 102.

Claims 1-20, 25, 27, 29, 31-34, and 38 were rejected under 35 USC § 103.

Summary of Response

Claims 3, 5, 7-9, 19, 22, 23, 25, 28, 30-34, and 36-38 are currently amended.

Claims 1, 2, 4, 10-16, 18, 21, and 35 were previously presented

Claims 6, 17, 20, 24, 26-27, and 29 remain as originally filed.

Summary of Claims Pending:

Claims 1-38 are currently pending following this response.

Response to Rejections

Nace, Sridharan, Tao-1, Tao-2 and Kashima do not disclose, teach or suggest non-intrusively monitoring a software application during real-time operation, whereby the application creates a shared memory area, stores application values in the shared memory area, and a monitor that attaches to the shared memory area and reads the stored application values without interfering with the operation of the software application involved. Generally, it is useful to monitor software (e.g., applications, programs, subroutines, modules, etc.) during normal operations. For example, conventional software applications may write intermediate information to certain internal variables, but the content of these intermediate variables is not output by the applications. Consequently, this content is not accessible to conventional software test or development programs without changing or encapsulating the software application involved. Therefore, it is more difficult for designers to develop, test or debug software applications using conventional techniques, without knowledge of the intermediate variables that were written as the software applications evolved. However, using one or more of the non-intrusive monitoring approaches disclosed in the pending application, a software application may be programmed to create a block of shared memory that can be shared by a software monitor. The software monitor may read application values that were written into the shared memory by the software application involved. Thus, for example, a developer may view an application's intermediate variables in order to understand and correct the operation of the application during its testing phase, or a business analyst may access the internal variables of multiple software applications in order to optimize the interactions between the multiple applications involved.

The pending application discloses systems and methods for non-intrusively monitoring a software application during real-time operation. The software application creates a shared memory area to be used for monitoring, and writes application values into the shared memory area while also performing allocated software tasks. A monitor module, which attaches to the shared memory area, reads the written application values while the software application is running in **real-time**.

Thus, the monitor module attaches to the shared memory area during normal operation of the software application, which enables the monitor module to access all of the internal variables, intermediate and otherwise, that the software application writes to the shared memory area. Therefore, the monitor module may access internal variable values of the software application that are otherwise not accessible by conventional testing or development techniques without changing or encapsulating the software application involved.

Nace is directed to an interprocess communications platform that enables individual processes to request and exchange data in a shared memory space, mediated by a communication engine. The communications engine locks a portion of the shared memory space allocated to a target process to permit a requesting process to access the data contained therein. Each process and the communication engine may create a shared memory space to be accessed by other processes within the system. However, real-time access to shared data is not provided because the communications engine (or other component) locks a portion of shared memory belonging to a target process, when another process requests data from the target processes share memory, thereby preventing the target process from performing temporarily while the requesting process accesses the shared data.

Sridharan is directed to a proposal for a non-intrusive framework for collecting performance statistics of Common Object Request Broker Architecture (CORBA)-based distributed systems. Sridharan describes an application of its non-intrusive framework to a large telecommunications system and the experimental results obtained. Specifically, Sridharan's study describes use of a Visibroker Object Request Broker (ORB) deployed to servers in the telecommunication system involved. The Visibroker ORB loads a performance instrumentation module together with a server into a single address space. According to Sridharan, sharing of the address space by an instance of the performance instrumentation module helps in collecting server specific information such as the CPU time, memory utilization, and the number of ORB threads launched. In other words, Sridharan generally describes a server and performance instrumentation module that shares an address space so the performance instrumentation module can retrieve performance details for the particular server involved.

Tao-1 is directed to a monitoring framework that allows users to understand the memory behavior of parallel applications. Specifically, Tao-1 discloses use of a hardware monitor as part of an event-driven monitoring approach that provides a user with detailed information about low-level memory transactions in multiple applications. Based on the statistics of the observed memory transactions, data or threads can be redistributed or reallocated more rationally among processors in order to reduce remote memory accesses and improve parallel performance. Tao-1 generally describes certain shared memory programming models, such as Pthreads, Win32threads, and SPMD, which partition the shared data of an application into 4096 byte pages that are distributed transparently among nodes. However, Tao-1 discloses no particular pertinent details about memory sharing or how it is performed.

Tao-2 is directed to a technique for visualizing the memory access behavior of shared memory applications on Non-Uniform Memory Access (NUMA) architectures. Specifically, Tao-2 discloses a visualization tool that displays monitored data in a user understandable way, in order to show the memory access behavior of shared memory applications. According to Tao-2, such tools require detailed information about the low-level memory transactions in the running program, and the only way to facilitate this without causing a high probe overhead is to perform the data acquisition using a hardware monitor capable of observing all memory transactions performed across the interconnection fabric. Also according to Tao-2, the disclosed hardware monitor traces memory references performed by the running program and gives the user a complete overview of the program's memory access behavior in the form of access histograms. However, similar to Tao-1, Tao-2 discloses no particular pertinent details about memory sharing or how it is performed.

Kashima is directed to an approach for constructing Web-based enterprise systems with distributed object technology. Kashima concentrates on issues to be considered while investigating the applicability of using distributed object technology in the Internet environment. Specifically, Kashima discusses the CORBA standard and maintains that with CORBA, connectivity with current systems is kept high because of CORBA's mainframe and COBOL support. Thus, Kashima teaches the use of CORBA for constructing the Web-based enterprise systems involved. Kashima mentions the use of unique name spaces in the Web environment, but Kashima does not disclose or suggest any details related to memory sharing or how it is performed.

As shown above, none of the above-cited art in the Office Action discloses, teaches or suggests non-intrusively monitoring a software application during real-time operation, whereby the application creates a shared memory area, stores application values in the shared memory area, and a monitor that shares the memory area reads the stored application values without interfering with the normal operation of the software application, as claimed.

This distinction, as well as others, will be discussed in greater detail in the analyses of the pending claims that follow.

Response to Rejections under 35 U.S.C. § 102

Claim 21:

Claim 21 was rejected under 35 U.S.C. § 102(e) as being anticipated by Nace et al, U.S. Pub. No. 2004/0268363 ("Nace").

I. Nace does not disclose obtaining a value for one or more of the plurality of variables written to the address space by the application during the **real-time** operation of the application.

Claim 21 of the pending application recites in part "the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application using the offset." In other words, a system according to claim 21 of the pending application allows variable values of an application to be non-obtrusively monitored in **real-time**. No interrupts are required to be inserted into the application code nor are any conditions placed on the application's ability to manipulate its data. Rather, the module merely attaches to the memory of the application and occasionally passively obtains values for one or more variable values. However, Nace discloses a system of shared memory resources in which a portion of shared memory space allocated to one process is locked to permit a requesting process to access data. (See, for example, Abstract of Nace). Thus, the operation of the target process is interfered with by the sharing mechanism since the target process is prevented from manipulating its data as it needs to in order to allow another process to access the data. This results in the operation of the target process being interrupted, therefore, any values for variables obtained by the requesting process is not performed in real-time, but, rather, requires that the other process be temporarily interfered with or even halted while the requesting process accesses the target processes data.

II. Nace does not disclose "attaching to an address space."

Claim 21 of the pending application recites in part "the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application using the offset." The module does not create a shared memory space, but merely attaches to the memory created by an application. Furthermore, the module does not write to the memory, but only reads the contents of the memory.

The Office Action alleges that Nace provides an application and a module and points to figure 2 as disclosing this teaching. The Office Action equates one of processes 102a and 102b with the application as recited in claim 21 and communication engine 108 with the module as recited in claim 21. However, the communication engine 108 in Nace does not attach to the shared memory space 104, but rather is able to create shared memory blocks and populate memory blocks for communication with other processes that are sharing the memory space. (See, for example, Nace, paragraphs 24, 25, 33, and 35). The communication engine 108 does not merely passively attach to an address space. Rather, the communication engine 108 can lock the shared memory block, create new shared memory blocks, and write to shared memory blocks. (See, for example, Nace abstract and paragraphs 19 and 24).

III. Nace does not disclose using the offset to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application.

Claim 21 of the pending application recites in part "attaching to an address space used by the application during real-time operation to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application using the offset." Nace does not disclose using the offset to obtain a value for a variable written to the address space. A text string search of Nace for "offset" produced exactly two results. One occurrence is in paragraph 22 and the other is in paragraph 39. In paragraph 22, Nace states that "[o]ther data 116 related to the newly-generated shared memory block in the set of memory blocks 114a, 114b . . . may also be loaded into administrative memory 106, which data may include for instance file handles, variables, address offset or other memory mapping data and other information related to the requesting process in the set of processes 102a, 102b . . . and its corresponding memory block." Thus, in paragraph 22, Nace merely discloses that an "address offset" may be part of the data loaded into administrative memory 106, but does not disclose how the offset may be utilized. In fact, the offset is not even required to be stored, but is merely an example of data that may be stored into administrative memory 106. In paragraph 39, Nace states that "[w]hen a memory block with the set of memory blocks 114a, 114b . . . or a segment thereof is released, the buddy of that block may be efficiently recaptured as well, by computation of the offset or other known relation to its associated buddy block or segment." It is clear that the offset is not used to determine an address space to obtain a variable written to the address space by an application, but is rather used to determine the address of another memory block that is related (e.g. buddy block) to a released memory block so that the "buddy block" may also be released or recaptured by the system.

For at least the reasons established above in sections I-III, Applicant respectfully submits that independent claim 21 is not anticipated by Nace and respectfully requests allowance of this claim.

Claims Depending from Claim 21:

Claims 22-24, 26, 28, and 30 were rejected under 35 U.S.C. § 102(e) as being anticipated by Nace.

Dependent claims 22-24, 26, 28, and 30 depend directly or indirectly from independent claim 21 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-III above, Applicant respectfully submits that claims 22-24, 26, 28 and 30 are not anticipated by Nace and respectfully requests allowance of these claims.

Claim 35:

Claim 35 was rejected under 35 U.S.C. § 102(e) as being anticipated by Nace.

Claim 35 includes limitations substantially similar to the limitations discussed in section I above. For example, claim 35 recites "a COBOL monitor module stored on a computer-readable medium that shares the shared memory area with the COBOL program through the technical layer, and the COBOL monitor module reads the program values stored in the shared memory area by the COBOL program during real-time operation of the COBOL program." Accordingly, the arguments of section I are hereby repeated for claim 35.

IV. Nace does not disclose COBOL.

Claim 35 recites in part a "COBOL program" and a "COBOL monitor." Nace does not disclose COBOL. The Office Action alleges that paragraph 28 of Nace discloses COBOL. However, paragraph 28 of Nace merely states that "it will be appreciated that different types of code or instruction may be used." Nace does not specify that COBOL is one of the different types of code that may be used. A text string search for "cobol" in Nace produced no results.

V. Nace does not disclose a technical layer.

Claim 35 recites in part that "a COBOL program stored on a computer-readable medium that creates a shared memory are through a technical layer" and "a COBOL monitor module stored on a computer-readable medium that shares the shared memory area with the COBOL program through the technical layer." COBOL applications generally cannot create or access shared memory areas. The technical layer provides a mechanism for the COBOL program to create a share memory area and for the COBOL monitor module to access the shared memory area. Nace does not disclose a technical layer. The Office Action alleges that the memory manager 118a or 118b in Nace is a technical layer. However, the memory manager 118a or 118b does not provide a mechanism for allowing COBOL programs and modules to create or access share memory areas. In fact, as stated above, COBOL is not disclosed in Nace at all.

For at least the reasons established above in sections I, IV, and V Applicant respectfully submits that independent claim 35 is not anticipated by Nace and respectfully requests allowance of this claim.

Claims Depending from Claim 35:

Claims 36 and 37 were rejected under 35 U.S.C. § 102(e) as being anticipated by Nace.

Dependent claims 36 and 37 depend directly or indirectly from independent claim 35 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I, IV, and V above, Applicant respectfully submits that claims 36 and 37 are not anticipated by Nace and respectfully requests allowance of these claims.

<u>Response to Rejections under 35 U.S.C. § 103</u>

Claim 1:

Claim 1 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan et al, *On Building non-Intrusive Performance Instrumentation Blocks for CORBA-based Distributed Systems,* March 2000, IEEE ("Sridharan").

Claim 1 includes limitations substantially similar to the limitations discussed in sections I and II above. For example, claim 1 recites "a first module stored on a computer-readable medium that shares and attaches to the shared memory area that is used by the at least one application during real-time operation, the first module reads application valued from the shared memory area that have been stored in the shared memory area by the at least one application during real-time operation." Accordingly, the arguments of sections I and II are hereby repeated for claim 1. Sridharan fails to cure the deficiencies in Nace identified and discussed above in sections I and II.

For at least the reasons established above in sections I and II, Applicant respectfully submits that independent claim 1 is not taught or suggested by Nace in view of Sridharan and respectfully requests allowance of this claim.

Claims Depending from Claim 1:

Claims 2-4 and 6-11 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan.

Claim 5 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan and further in view of Hiroshi Kashima, *An Approach for Constructing Web Enterprise Systems on Distributed Objects*, Jan., 2000, IBM ("Kashima").

Dependent claims 2-11 depend directly or indirectly from independent claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I and II above, Applicant respectfully submits that claims 2-11 are also not taught or suggested by Nace in view of Sridharan and respectfully requests allowance of these claims. Applicant respectfully submits that Kashima does not cure the deficiencies of Nace in view of Sridharan as noted above.

Claim 12:

Claim 12 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan.

Claim 12 includes limitations substantially similar to the limitations discussed in sections I-III above. For example, claim 12 recites "running an application in a real-time manner," "writing, by the application, the application values in the memory area during the operation of the application," and "reading, by a monitor, the memory area used by the application to obtain the application values." Accordingly, the arguments of sections I and II are hereby repeated for claim 12.

For at least the reasons established above in sections I and II, Applicant respectfully submits that independent claim 12 is not taught or suggested by Nace in view of Sridharan and respectfully requests allowance of this claim. Sridharan does not cure the deficiencies in Nace.

Claims Depending from Claim 12:

Dependent claims 13-20 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan.

Dependent claims 13-20 depend directly or indirectly from independent claim 12 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I and II above, Applicant respectfully submits that claims 13-20 are not taught or suggested by Nace in view of Sridharan and respectfully requests allowance of these claims. Sridharan does not cure the deficiencies in Nace.

Claims Depending from Claim 35:

Dependent claim 38 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Sridharan.

Dependent claim 38 depends directly or indirectly from independent claim 35 and incorporates all of the limitations thereof. Accordingly, for at least the reasons established in sections I, IV, and V above, Applicant respectfully submits that claim 38 is not taught or suggested by Nace in view of Sridharan and respectfully requests allowance of this claim. Sridharan does not cure the deficiencies in Nace.

Claims Depending from Claim 21:

Dependent claims 25 and 31-32 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Jie Tao et al, *Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures,* Springer-Veriag Berlin Heidelberg 2001, pp. 861-870 ("Tao-2").

Dependent claims 27 and 29 were rejected under 35 U.S.C. § 103(as) as being unpatentable over Nace in view of Huang et al., *Operating System Support for flexible Coherence in Distributed Shared Memory,* 1996, IEEE ("Huang").

Dependent claims 33-34 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Nace in view of Tao-2 and further in view of Tao et al, *Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework,* March 2000, IEEE ("Tao-1").

Dependent claims 25, 27, 29, and 31-34 depend directly or indirectly from independent claim 21 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-III above, Applicant respectfully submits that claims 25, 27, 29 and 31-34 is not taught or suggested by Nace in view of Tao-2 and respectfully requests allowance of these claim. Neither Huang nor Tao-1, alone or in combination, cure the deficiencies of Nace in view of Tao-2.

CONCLUSION

Applicant respectfully submits that the present application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, the Examiner is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: January 2, 2009

/Michael W. Piper/
Michael W. Piper
Reg. No. 39,800

Conley Rose, P.C.
5601 Granite Parkway, Suite 750
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)

ATTORNEY FOR APPLICANT